Semantic Segmentation with Edge Supervision

Feng Ling
Harvard T.H. Chan School of Public Health
665 Huntington Ave, Boston, MA 02115
lingfeng@hsph.harvard.edu

Kexin Yang
Harvard T.H. Chan School of Public Health
665 Huntington Ave, Boston, MA 02115
kexin_yang@hsph.harvard.edu

Abstract

Improving boundary segmentation has been a concern for semantic segmentation tasks for a long time. In this project, we propose three new deep learning models with edge supervision for pixel-wise segmentation[1]*. The main idea of our proposed architectures includes a Segmentation Net for segmentation prediction and three variants of edge supervision methods, and they are combined to give the final segmentation result. The loss function is a weighted sum of segmentation prediction loss and edge classification loss, and the gradient will back-propagate through the Segmentation net and Edge net correspondingly. The three proposed architectures share the same idea but have difference in implementation details. The U-Net model is used as a baseline for comparison.*

All models were trained on $1/2$ *of the ADE20K Dataset for 30 epochs with the same hyper-parameter. The experiment results indicate that our proposed models have better performance on all three measures including pixel accuracy, top 50 IoU, and weighted IoU. We also noticed that each model achieved highest performance in one of the three measures, which suggests that different methods of edge supervision allows the models to learn different types of object segmentation better. This is also supported by our analysis of model IoU across 151 individual object classes.*

1. Introduction

Semantic segmentation has long been a hot topic for research, its applications ranging from instance segmentation, scene parsing and autonomous driving. Early methods mostly employed region proposals and engineered features to do patch-wise classification [11] [10]. Nowadays most of the modern methods relied on deep neural networks including FCN [8], SegNet [1], U-Net [9] and tranformer based U-Netr [5]. The segmentation results of these models seem still unsatisfying due to the loss of edge information during encoding image features [1]. Some of the CNN models make use of layer fusion and feature concatenation [8] [9] to improve boundary localization, but didn't do well for predicting small objects [1]. Other improvements resides in applying edge filters and treat the image body (low-frequency signals) and the edge (high-frequency signals) seperately [7].

Here we proposed three model architectures to improve the boundary localization based on the U-Net architecture. These architectures have a Segmentation Net for semantic segmentation, and different variants of edge supervision methods. We also proposed a loss function which is a weighed sum of segmentation loss and edge classification loss. These architectures forces the U-Net model to pay more attention to the edge pixel classification. The first architecture treats edge prediction as binary classification task, while the second model treats it as multi-class classification. The third model makes use of pre-trained Edge Net to guide the segmentation task. We compared the performance of U-Net and our proposed models based on pixel accuracy, top 50 mean IoU, weighted IoU, and visualization to evaluate the effectiveness of edge supervision on semantic segmentation task.

We trained our models on MIT ADE20K scene parsing dataset for both indoor and outdoor semantic segmentation [13]. Some of the real images and the ground truth annotations are shown in Figure 1. We did not aim to achieve SOTA performance in this project due to limits in time and computational resource. Instead, our main goal was to explore the effect of adding special attention to edge on the performance of semantic segmentation by comparing the performance of our proposed architectures

[1] You can find relevant codes on our Github repository: https://github.com/fengling0410/Advances-in-Computer-Vision-Final-Project-Semantic-Segmentation-with-Edge-Supervision

with the baseline U-Net.

2. Related Work

2.1. Region Proposal and Sliding Window

Before neural networks, the prevalent methods were regional proposals (such as pixel clustering) and sliding window, as in object detection. Image patches were thrown into a classifier such as Random Forest [10] or Boosting [11]. When neural network first came into play, semantic segmentation used neural network as the classifier but still relied on region proposals to enable training. Obviously, these methods were slow (mainly due to region proposals) and led to a trade off between size of image patches and segmentation accuracy [9].

2.2. Fully CNN-based Model

The Fully Convolutional Network (FCN) [8] was the first segmentation method that only maked use of convolutional neural network. It built "fully convolutional" networks that take input of arbitrary size and produce output of the same size. The architecture of FCN comprised of a down-sampling encoder followed by an up-sampling decoder. The encoder was built on the intuition that important information about the input image can be maintained using embeddings. However, during the down-sampling process, the would be localization information loss since the whole image was represented with embeddings that had much smaller dimensions. The paper proposed layer fusion which skip connects the de-convolution layer with feature maps from down-sampling process. This layer fusion method increased the performance of FCN, but had a diminishing return, and no improvement could be seen beyond 3 layers.

Another fully convolutional segmentation model was U-Net [9] which was primarily designed for biomedical image segmentation. It added a concatenation connection between up-sampling layer and the feature maps extracted from down-sampling process, thus providing localization information for up-sampling process. This architecture worked well for small annotated dataset. The paper also proposed adding weights to cell boundary pixels to improve boundary segmentation for pathology images.

After FCN and U-Net, Segnet [1] was proposed mainly for road scene understanding and indoor scene segmentation. It was inspired by UNET and has a similar architecture with encoder, decoder, and skip connection. The difference between Segnet and U-Net was that only the pooling indices were transferred to the expansion path in Segnet, instead of the whole feature maps as in U-Net. Therefore, Segnet consumed less memory compared with Segnet.

The Segnet paper has shown that when applied to indoor scene that contains lots of small objects and appear infrequently, most of the current segmentation neural nets architecture have poor performance, suggesting a room for improving for CNN based segmentation models [1].

2.3. Transformer

Transformer was originally proposed for natural language modeling and has soon become the state-of-art in language modelling [12]. Dosovitskiy at al. [4] applied transformer for image recognition at scale, and it achieved higher accuracy than Resnet [6] which had the highest accuracy at that time when the model was trained on large amount of image dataset. Convolutional neural network naturally introduces inductive priors: pixel only cares about its neighbourhood, while the relationships between image patches are learnt on the fly. This less biased intuition makes vision transformer works better than Convolutional neural networks on large dataset.

UNETR [5] is a segmentation architecture similar to UNET but uses vision transformer as the feature extractor. It has similar encoder, decoder, and feature concatenation as UNET with encoder replaced by vision transformer. It has achieved the highest accuracy on BTCV dataset compared with other state-of-the-art CNN based models and has become the new state-of-the-art in the transformer era.

2.4. Edge Supervision

Many previous works have been done to improve the edge predictions in semantic segmentation model. These research mainly focused on improving the smoothness and accuracy of the predicted edge, and several different approaches have been proposed and shown to effectively improve the semantic segmentation outcome.

In the work by Chen et al. [3], they proposed a method to improve the poor localization of object boundary with Deep Convolutional Neural Networks by combining the response of DCNN with a fully-connected Conditional Random Field. This methods fine-tuned the segmentation score predicted by DCNN using the more accurate edge information obtained from the fully-connected CRF, which is a computationally expensive process. They later proposed another method [2] that first use a CNN to predict both image segmentation score and edge, and then applied

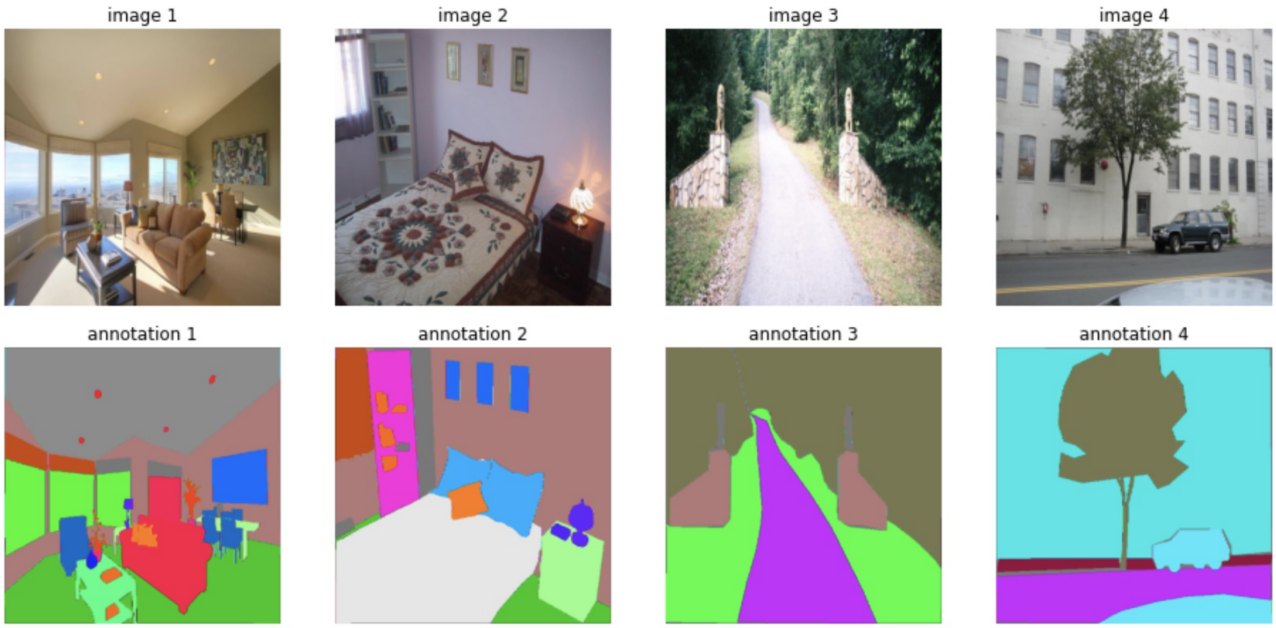


Figure 1. ADE20K Scene Parsing Images and Corresponding Ground Truth Annotation

domain transform to filter the segmentation score and align with object boundaries. This architecture allows the model to directly learn to optimize semantic segmentation with object edge boundaries through back-propagation.

Another model [7] decoupled the semantic segmentation into two separate task that focus on explicitly learning the object body and edge. They generated body and residual edge annotation by splitting the high frequency and low frequency contents of the image, and then they supervise the body feature and edge feature learning separately, in the body generation module and edge preservation module respectively. Finally, body and edge feature features were merged together to generate the final prediction, and the final loss function was a weighted sum of body loss, edge loss, and a final loss.

3. Approach

Inspired by the work decoupled semantics segmentation model that learns edge and object separately [7], we proposed a method to supervise semantic segmentation with an attention on object edges. Our main goal here was to test the effect of a single edge supervision on image segmentation task, and to see whether and how much it can improve the performance of a image segmentation model. Here we chose U-Net [9] as our baseline model, and we built 3 models that incorporate variations of edge supervision methods based on the original U-Net structure.

All the models were trained for 30 epochs using an initial learning rate of 0.001. The optimizer used is Adam for all the models. Due to computation limit, all the models were trained and validated on $1/2$ of the ADE20K dataset.

3.1. Model 1: SegNet With Binary Edge Supervision

We designed Model 1 as a vanilla model based on the UNet [9] structure with simple binary edge supervision. The structure for Model 1 is shown in Figure 2. The binary edge annotations were obtained by applying Laplacian filter $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ to the image segmentation annotation provided in the Dataset and converting the results to binary matrices to indicates either edge or not edge. Our intuition for using the binary edge annotation was that we hope the model can learn to distinguish the important edges that separates the objects in the image, and thus use these additional knowledge about object edge to aide the image segmentation task.

To train model, a batch of images was first passed into the U-Net, which output a segmentation prediction of size $bs \times 151 \times h \times w$ and an edge prediction of size $bs \times 1 \times h \times w$. The segmentation loss was calculated as the cross entropy between segmentation prediction and annotation to measure the amount of difference in each object class. The edge loss was calculated using binary

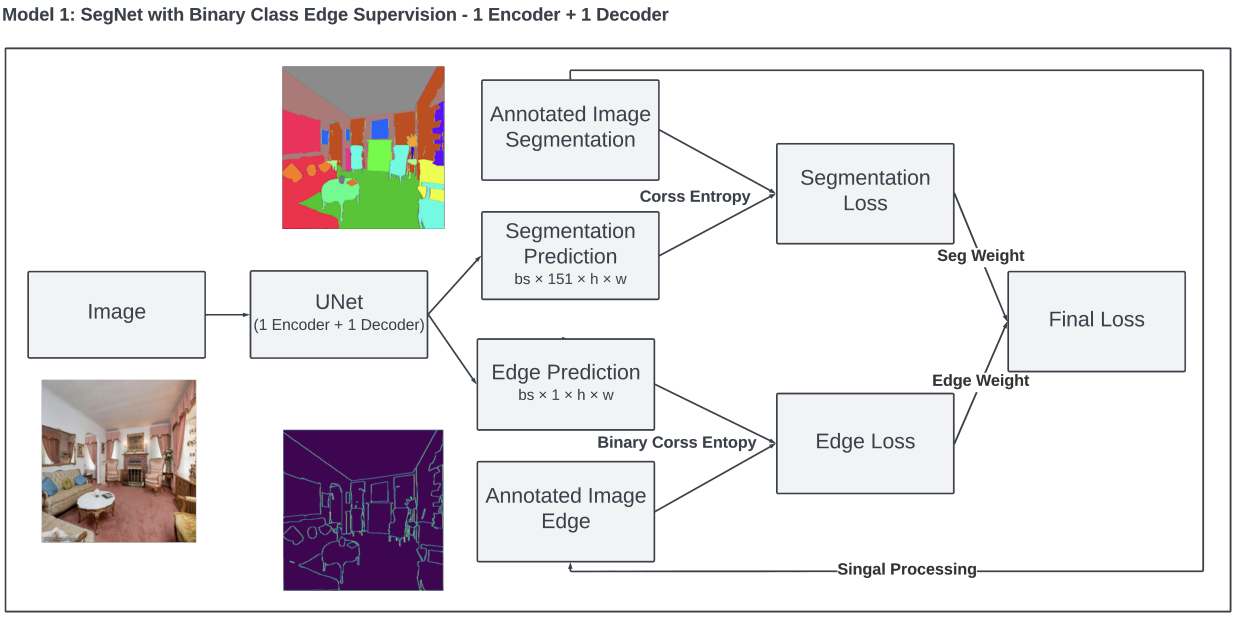


Figure 2. Structure for Model 1. Image segmentation based on UNet with binary edge prediction as supervision.

cross entropy, which simply measured the amount of difference across all object classes. The final loss was the weighted sum of segmentation loss and edge loss. The Adam optimizer was applied to update the model weight based on the final loss to allow the learning of segmentation and edge prediction simultaneously with different importance as specified by the weights.

3.2. Model 2: SegNet With Multi-Class Edge Supervision

After visualizing the output from our vanilla Model 1, we discovered that the edge prediction output by the model was, in some extend, similar to the results of simply applying edge detection on the original image. For instance, we saw the edge prediction contains edges inside the objects as well, which is irrelevant for the image segmentation task. This differed from our expectation for the model to learn to identify only the important edges, and the additional edges inside the objects can potentially add negative interference. To account for this issue, we then proposed Model 2 with an improved structure shown in Figure 3, which allowed the model to learn about the important object edges better.

There were two main improvements for Model 2. First, we decided to apply multi-class edge supervision instead of binary class to provide more information to the model. The multi-class image edge annotation was obtained by applying a contour filter $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ to each object class of the image segmentation annotation separately, and combining the results together where the pixel value of the edge (1 - 150) indicates either the object class or not edge (0). So the colour of the edge would correspond to the colour of the object in the original segmentation annotation.

Second, We revised the U-Net structure to include 2 encoders so that the segmentation and edge feature encoding can be learned with separated weights. Resnet34 was used to extract image features and Resnet18 was used to extract edge features. Image features and edge features were then concatenated and decoded through one decoder to allow the model learn to incorporate these knowledge together. The model output segmentation and edge predictions separately, both of size $bs \times 151 \times h \times w$, and cross entropy losses for segmentation and edge were calculated respectively and summed together with corresponding weight to produce the final loss. A class weight with small weight for not-edge class (0) and large weight for all other object classes (1 - 151) was used for edge loss in order to prevent the model from making only not-edge prediction and thus forcing it to learn to predict edges for each object class. Here we used a naive class weight that contained 0.01 for not-edge class and 1 for all object classes. For further improvement, class weight can be built based on the actual class ratio in the training Dataset to allow better more balanced learning

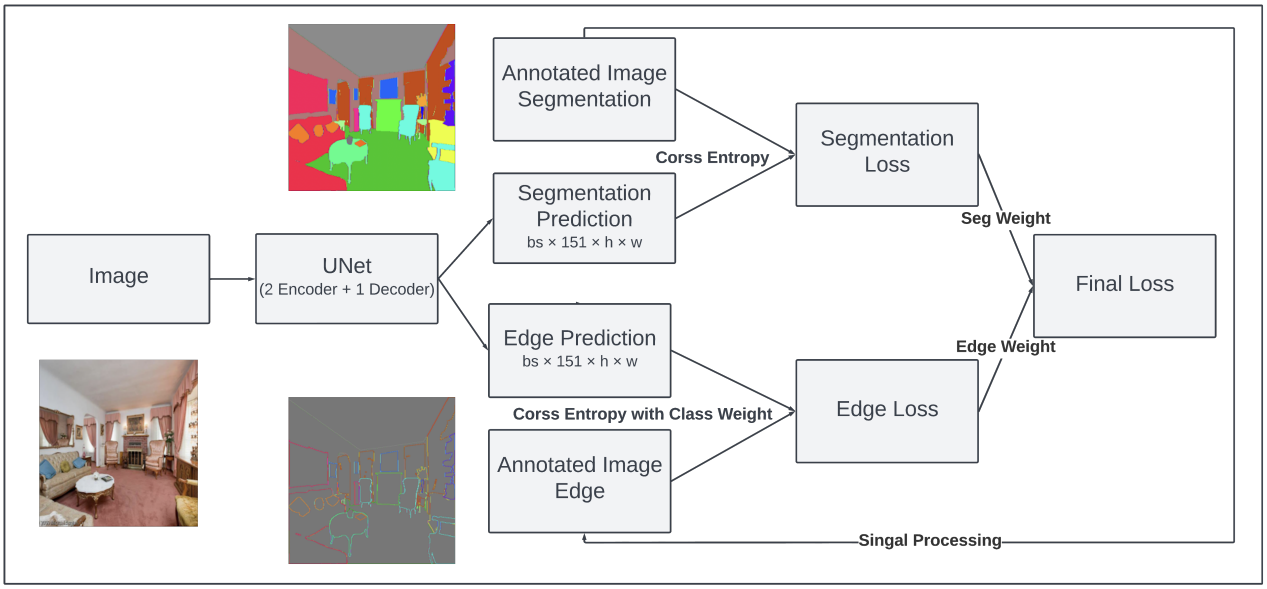


Figure 3. Structure for Model 2. Image segmentation based on UNet with multi-class edge prediction as supervision.

across different object classes.

3.3. Model 3: SegNet With Pretrained EdgeNet Supervision

We observed improved edge prediction from model 2, which suggested that the encoded edge features can likely provide additional information that can help model with better segmentation prediction. However, since the model still shared the decoder weight, we cannot directly measure how well the edge encoder learned based on visualization of the edge prediction outcome since this outcome also contained segmentation feature encoding.

In order to directly visualize the model's edge prediction and guarantee that it learns to extract different classes of object edge, we need to train a model specifically for edge prediction task. So we constructed Model 3 as two separate parts – a pre-trained EdgeNet and a SegNet as shown in Figure 4.

The EdgeNet was trained by passing the image into a UNet, which output edge predictions of size $bs \times 151 \times h \times w$. The edge predictions were then compared with the multi-class image edge annotation obtained using the same methods as Model 2, and the edge loss was also calculated as the cross entropy with small class weight for not-edge class which force the model to learn to predict edge for object classes.

Both image and the encoded edge feature extract from the pre-trained EdgeNet encoder were used as the input for SegNet. Only the image was passed into the encoder of the UNet, and the encoded image feature was then concatenated with the encoded edge feature to pass into the decoder. Only a segmentation prediction of size $bs \times 151 \times h \times w$ was output from the model, and the segmentation loss was computed as the cross entropy loss between segmentation prediction and annotation.

For our Model 3, we decided to fix the weights of the pre-trained EdgeNet during the segmentation task for simplicity since the EdgeNet were able to achieve a relatively good edge prediction after the initial training.

4. Experimental Results

Figure 7 shows the first 4 images and the ground truth segmentation annotation in the ADE20K validation Dataset. In Figure 8, we showed the predictions for these 4 images by the baseline and the 3 models to visually compare and analyse their performance.

4.1. Baseline Model

The baseline model we have chosen is U-Net with pre-trained ResNet34 architecture as the encoder. The

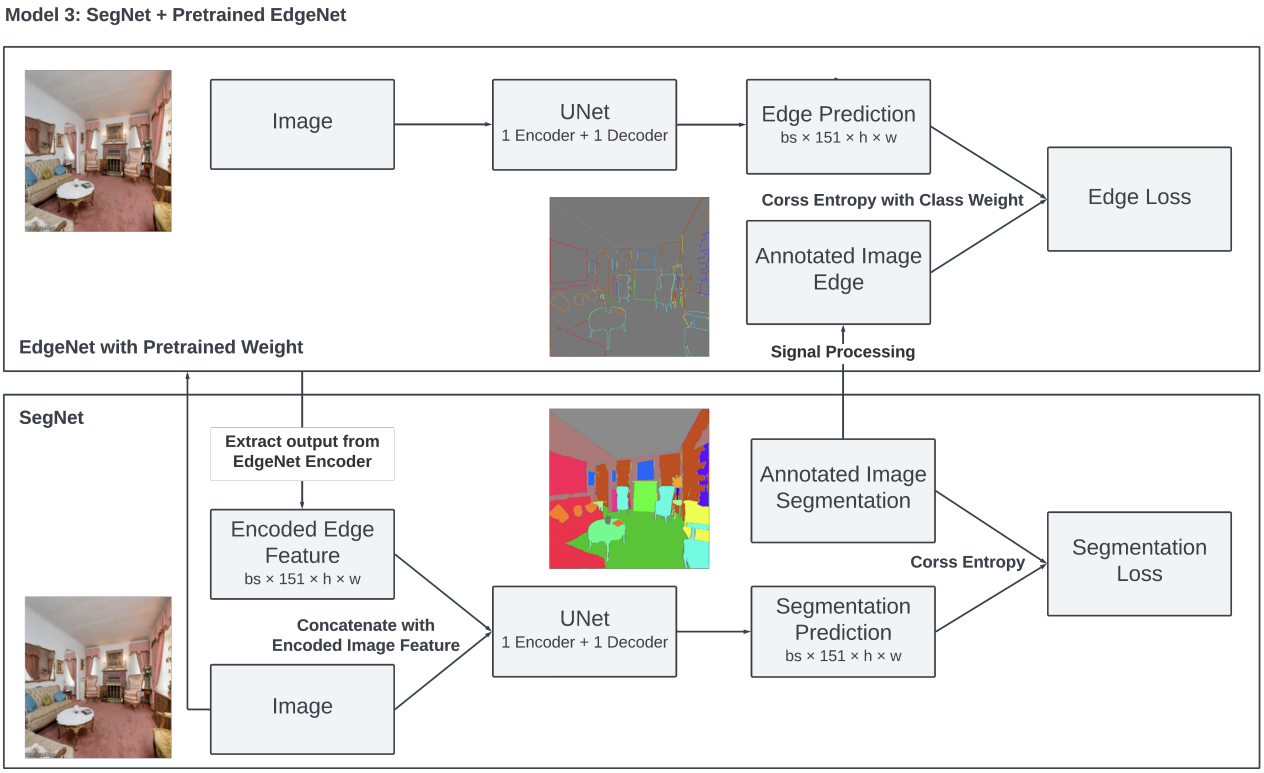
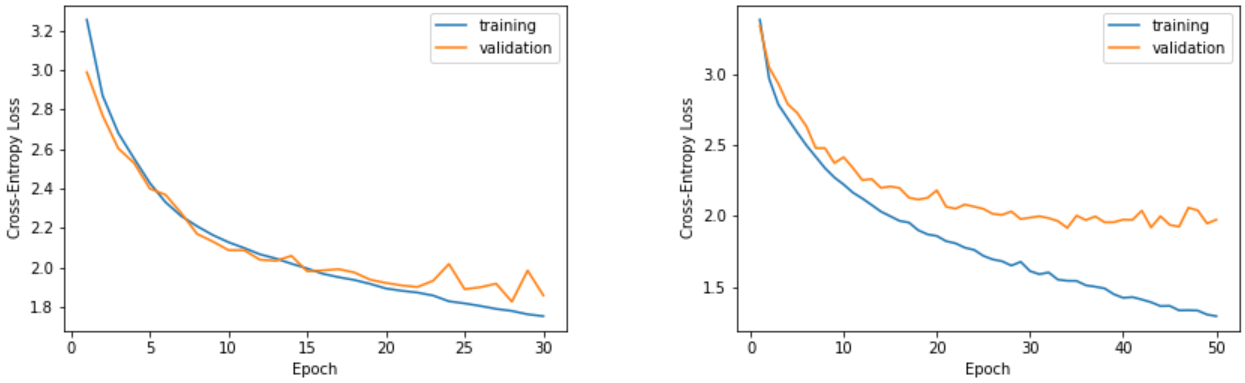


Figure 4. Structure for Model 3. Image segmentation based on UNet with pre-trained EdgeNet to output encoded edge feature.



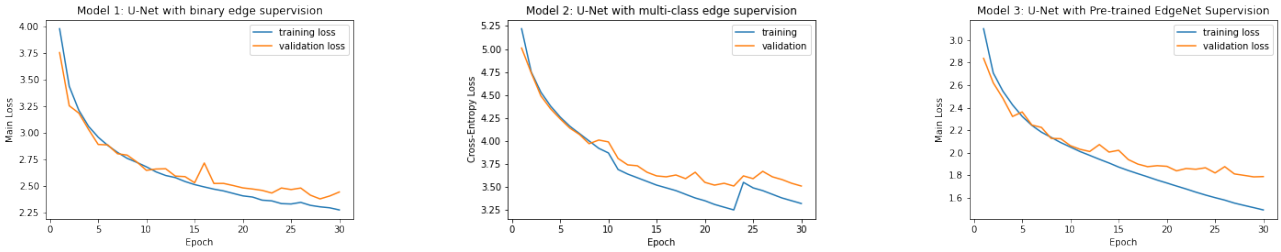
a Training on $1/2$ Dataset for 30 epochs

b Training on $1/5$ Dataset for 50 epochs

Figure 5. Train and validation loss for baseline U-Net model

ResNet34 encoder is pre-trained on ImageNet dataset. For the baseline model, filters size are set to be 512, 512, 256, 128, 64, corresponding to each convolution layer of Resnet34. The optimizer is set to be Adam with learning rate 10^{-3}. In consideration of memory limit, batch size is set to be 8. The loss function used for training is pixel-wise cross-entropy loss.

Figure 5 shows the training and validation loss of our baseline model. The loss plot indicates a good convergence and almost no over-fitting when using $1/2$ of the ADE20k as training Dataset. The pixel accuracy on the validation



a Model 1: Binary Edge Supervision

b Model 2: Multi-class Edge Supervision

c Model 3: Pre-trained EdgeNet

Figure 6. Train and validation loss for models after training on $1/2$ Dataset for 30 epochs

Model	Pixel Accuracy	Weighted IoU	Mean IoU 50
Baseline	55.5	38.45	17.66
Model 1	57.4	**41.41**	20.47
Model 2	**57.9**	38.5	18.46
Model 3	57.2	41.22	**22.15**

[1] Mean IoU 50: The Mean IoU of the top 50 most common classes in the ADE20k dataset. [2] Weighted IoU: This is the mean IoU weighted by class ratio. [3] The IoU data by class can be found in the Appendix.

Table 1. Summary of Evaluation Metrics for All the Models

Baseline	Model 1	Model 2	Model 3
3	24	14	32

Table 2. Count of Best IoU Prediction Class for All the Models

Dataset is 55.3%, the Mean IoU is 6.60%, the Top 50 IoU is 17.66%, and the weighted IoU is 38.45%. While the pixel accuracy and Mean IoU is much lower than the SOTAs on ADE20K, but considering the relatively short training time and small training set, its performance is still acceptable.

From the experiment with pre-trained U-Net, it is visible that larger Dataset leads to lower training and validation loss and less overfitting, see the loss plot of training on $1/5$ of the dataset for 50 epochs in Figure 5. For consistency and better comparison, we will take the U-Net model trained on $1/2$ of the ADE20K Dataset for 30 epochs as the benchmark for comparing with our proposed edge-supervised models.

We will compare our model performances with these baseline performance to evaluate the effect of edge supervision on semantic segmentation on $1/2$ of the ADE20K Dataset after 30 epochs of training. The comparison will be based on pixel accuracy, Mean IoU and segmentation prediction visualization. Specifically, whether edge supervision can help improve model performance on semantic segmentation, and if so, to what extend can different methods of edge supervision help improve the performance.

4.2. Model 1: Binary Edge Supervision

For hyper-parameters, we use similar settings as the pre-trained U-Net model with ResNet34 to be the encoder. The optimizer is Adam with learning rate 10^{-3} and default momentum. The loss here is defined to be a weighted sum of edge loss (binary cross-entropy) and segmentation loss (multi-class cross-entropy) $\lambda_{edge} \times Loss_{edge} + \lambda_{seg} \times Loss_{seg}$. Here we set λ_{edge} to be 0.3 and λ_{seg} to be 1. The loss plot is shown in Figure 6a, and we can see that there is almost no over-fitting.

The pixel accuracy of model 1 is 57.35%, the Mean IoU of top 50 classes is 17.66%, and the weighted Mean IoU is 41.41%. All of the metrics are higher than those of pre-trained U-Net.

The resulting segmentation and edge prediction plots from Model 1 are shown in Figure 8b. We can see from the edge prediction plot that, while it is similar to the result of simply apply edge detection filters on the image, the important edges that separate different object appear to be significantly bolder than the irrelevant edges inside the objects. This suggests that the binary edge encoding is still likely to contains information about object edges that can help the model on the segmentation task. This can also be observed on the segmentation prediction plots. We can see that Model 1 is able to segment context on the image that the U-Net baseline model fails to distinguish, such as the wall indicated by the bright green color in image 1, and it also segment the car better in image 4. We can also observe that these improved segmentation corresponds to strong predictions of edge, which support the theory that the edge encoding contribute to the improved performance for Model 1.

4.3. Model 2: Multi-Class Edge Supervision

For Model 2, we applied similar parameter settings, except we used a learning rate scheduler with a decay of 0.9 after every 2 training epochs for the Adam optimizer. The image encoder is ResNet34 and and the edge encoder is ResNet18. Here the loss function is defined to be a weighted sum of cross-entropy loss for multi-class edge classification and cross-entropy lass for multi-class segmentation. The loss weights are the same as that used in Model 1.

The loss plot of model 2 is shown in Figure 6b. There is an immediate increase in training loss at the epoch 25, which could possibly due to the fact that we restored the weight from epoch 24 to restart the training, but did not change the initial learning rate according to the decay rate. Model 2 was able to obtain slightly better validation loss after 5 epochs of training, but it should be pointed out that Model 2 can potentially achieve better performance without our mistake during training. The pixel accuracy of model 2 is 57.9%, the weighted Mean IoU is 38.5%, and the Mean IoU of top 50 classes is 18.46%. All of the metrics are higher than those of pre-trained U-Net. Model 2 has higher pixel accuracy than Model 1, but the weighted IoU and Mean IoU among top 50 classes are lower.

The resulting segmentation and edge prediction plots from Model 2 are shown in Figure 8c. The edge prediction plots indicate some level of edge learning, and the edge and segmentation prediction images also share high level of similarity, but at the same time, the segmentation prediction contains less noise than the edge prediction, as indicated by the better prediction for buildings in image 1 and 2. These observations agree with our design of the model, which contains two separated encoders for edge and segmentation but one shared decoder to enable better association of the edge and segmentation feature. One clear drawback of this design is that it probably increased the level of difficulty for the model to directly learn and optimize for edge prediction due to the sharing of the decoder. This is reflected in the edge prediction plots, where most of the predictions portraits segmentation instead of edge. This also prevents us from directly evaluating the model's edge prediction performance.

4.4. Model 3: Pre-trained EdgeNet Supervision

Model 3 has the similar hyper-parameter setting as previous models except also using a learning rate scheduler with a decay of 0.9 after every 2 training epochs for the Adam optimizer. The EdgeNet is pre-trained using ResNet34 architecture for 30 epochs on $1/2$ ADE20K dataset using a U-Net architecture, and its output from encoder is extracted as edge features. These extra features are concatenated with the encoder output from the SegNet and passed into the decoder. The weights of EdgeNet is freezed thought gradient flow. Only the weights of SegNet are trainable and updated during segmentation training. Since we are not training the Edge feature extractor, the loss function here is just the cross-entropy loss of multi-class segmentation.

Figure 6c shows the loss trace. There is evidence of over-fitting after 25 epochs. The pixel accuracy of Model 3 is 57.2%,the Mean IoU of top 50 classes is 22.15%, and the weighted Mean IoU is 41.22%. Model 3 has the highest mean IoU scores, but the pixel accuracy and weighted IoU is slightly lower than Model 2 and Model 1 respectively.

The resulting segmentation and edge prediction plots from Model 3 are shown in Figure 8d. Since the EdgeNet is pre-trained to specificity predict object edge and used the multi-class edge annotation created from the annotated segmentation, it was able to output the edges as its class number with a good level of accuracy. By comparing the segmentation and edge prediction plots, we can also observe that the model is mostly able to correctly predict the segmentation class if the edge prediction for the object is correct and continuous, however, if the edge prediction is wrong, then the segmentation prediction is likely to be wrong as well. For instance, we can see that the edge prediction for the pavement in image 3 and the airplane in image 4 are either discontinuous or incorrect, therefore, the segmentation prediction for these object also reflect the same mistakes.

We also observe that the segmentation prediction may contains additional errors, such as the dark blue prediction inside the building in image 1. We believe that adding additional object supervision that penalize prediction of different object classes may help address this problem. Another possible direction is to let the model predict object class based on the surrounding edge class. This is because the edge prediction after only 30 epochs of training is already pretty good, so if the EdgeNet can be tuned for further improvement, it is possible to directly "fill" the object by inferring its class based on the nearest edge predictions and obtain a good segmentation prediction.

5. Discussion and Conclusion

In this project, we have proposed three models with edge supervision and compared their performance with the baseline U-Net model. Table 1 shows the summary metrics of all the models. We evaluate each model based on pixel accuracy, weighted Mean IoU, amd Mean IoU

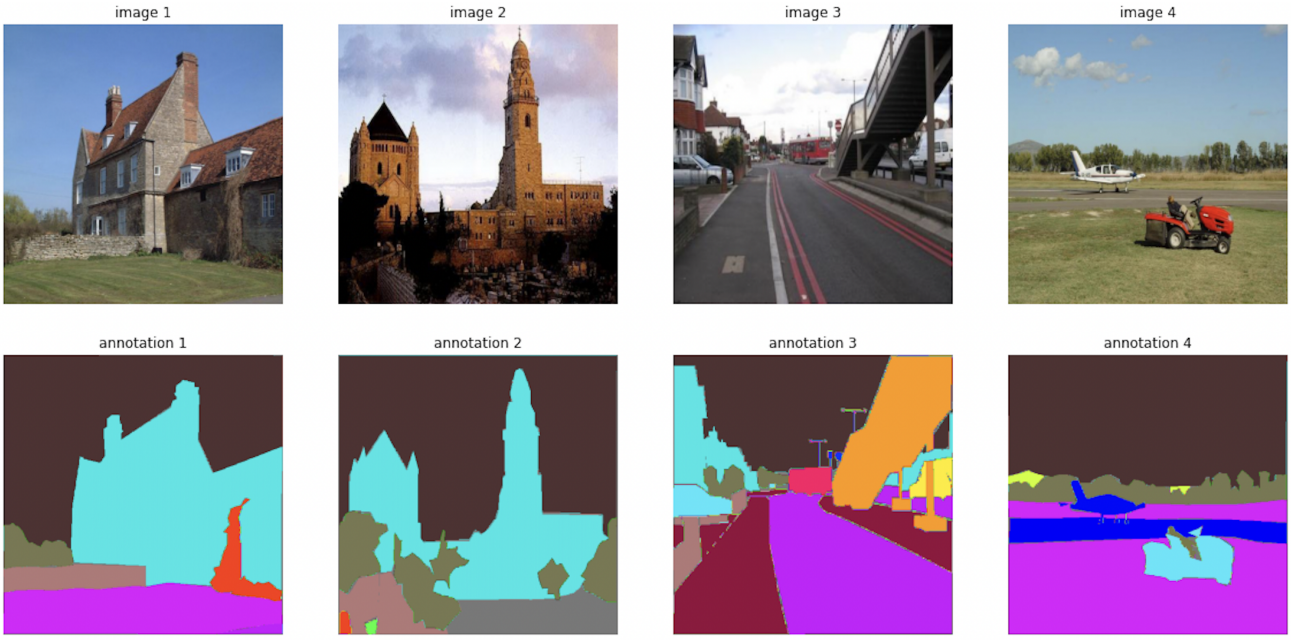


Figure 7. Ground Truth Image and Annotated Segmentation

of the top 50 most common classes in the ADE20K Dataset (Mean IoU 50). According to Table 1, all of our proposed models have better performance than the baseline pre-trained U-Net based on all the evaluation metrics. This result indicates that the information provided by edge supervision is valuable for segmentation task.

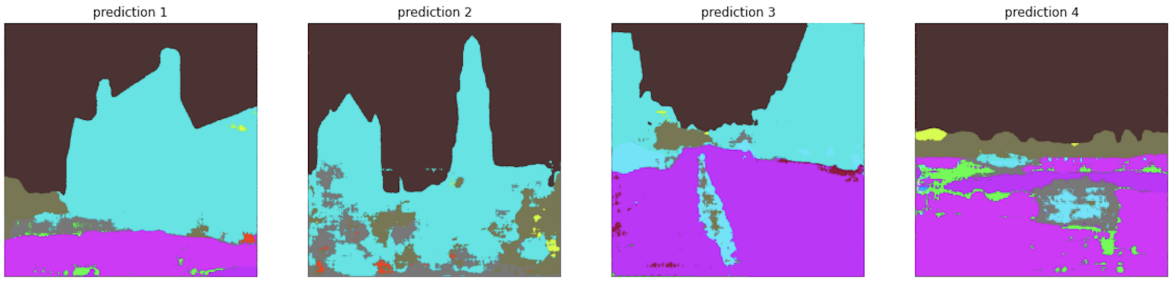
Among the three proposed models, Model 2 has the highest pixel accuracy, while Model 1 has the highest weighted IoU, Model 3 has the highest Mean IoU among the top 50 classes. However, there is no much visible difference of evaluation metrics between them. The loss plots show good convergence of all the models, and there seems to be some overfitting for Model 3 and Model 2. We hypothesis that using larger Dataset for training could mitigate the over-fitting problem.

Figure 8 is the visualization of segmentation/edge prediction given by all the proposed models. Model 1 and Model 2 have visibly better segmentation, especially on the object boundary, while the visualization of Model 3 is not that good. For edge prediction, the visualization of binary edge prediction is pretty good. Note that the learning of binary boundary prediction doesn't just means learning simple edge filters, it also learns the importance of different edges. The edge prediction in Figure 8c shows the prediction gives higher weights to those boundaries that are important for segmentation and ignores edges of high-frequency details.

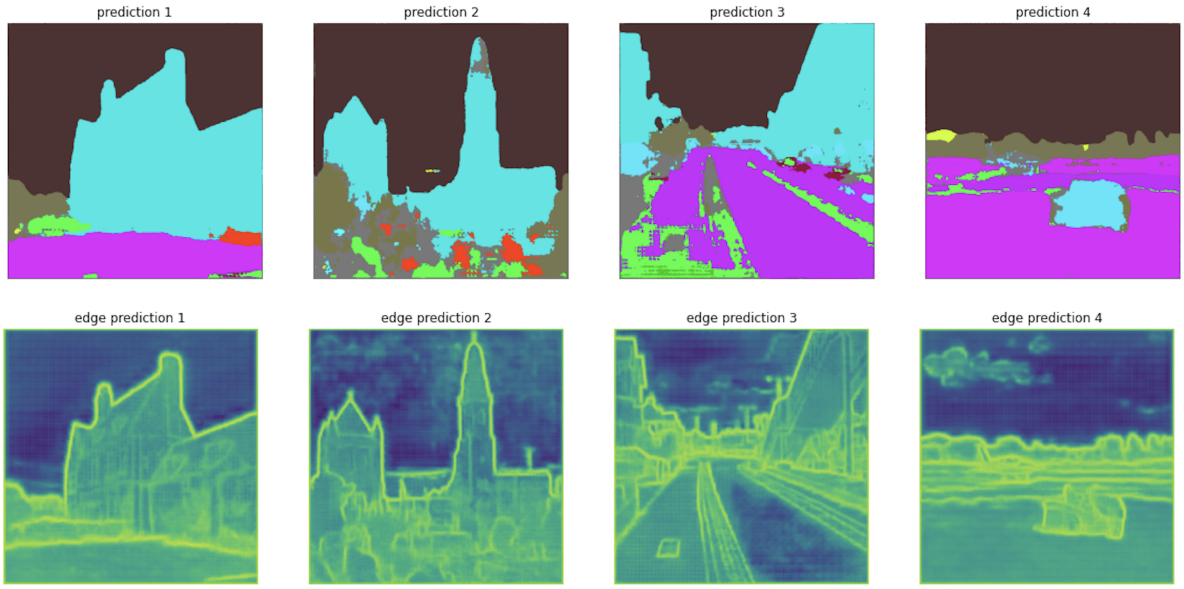
The multi-class edge prediction of Model 3 also looks reasonable. It has successfully recognized boundaries that are useful for segmentation. Notably, the segmentation has better performance where the edge prediction is correct, while it has bad segmentation prediction where the edge prediction is wrong, which shows the edge prediction has provided importance guidance to the segmentation task.

Since the 3 proposed models each shows best results in one of the 3 measures, we think it would also be interesting to look at the IoU performance for each individual class and see if the models may show differences in prediction accuracy across different classes. Table 2 shows a summary of the count of best non-zero IoU for each class for the baseline and 3 proposed models, and the individual IoU performance are shown in the Appendix Table. We can see from this table that each model did show different levels of prediction IoU for different classes, which suggests that these models may be good at learning different types of classes. For instance, we can see that Model 3 has a significantly higher class IoU for object class cabinet (class 11), table (class 16), and chair (class 20), which is 18.59%, 17.04%, and 28.52% higher than the second best IoU for that class. This suggests the structure for Model 3 may makes it better at segmenting furniture compared with the other models. On the other hand, Model 1 has

a Segmentation Prediction of Pre-trained U-Net



b Segmentation and Edge Prediction of Model 1



the highest IoU for classes such as sky (class 3), grass (class 10), and mountain (class 17), suggesting that it may be better at segmenting natural landscape than the other models. With this observation, we proposed that it may also be beneficial to build an ensemble model that use the votes from multiple segmentation models to obtain the final semantic segmentation, since each model may learn to segment some classes of objects better.

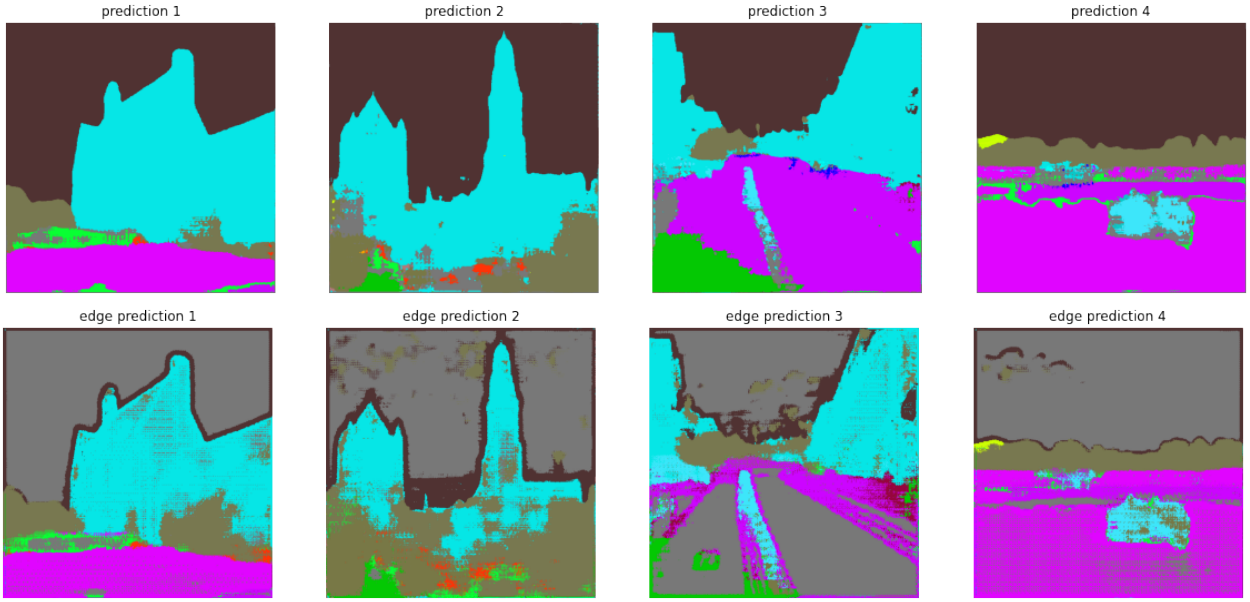
Another obvious observation from the individual class IoU table is that all models are bad at predicting object classes that appears less frequent in the Dataset. While this is expected, it also provides a direction for improvement. For instance, adding additional attention for these less frequent classes such as class weight could potential further improve the model performance.

In conclusion, we have have proposed three models with edge supervision to explore the influence of edge prediction on segmentation performance. Although these methods have different edge definition (binary or multi-class), feature concatenation methods (on the bottleneck layer or upper layer), encoder and decoder structure (number of encoders and decoders), their performances have surpassed the baseline U-Net. We are glad to see such an improvement with edge supervision, and we hope our project could provide insights to other researchers and ease further researches in this field.

6. Limitation and Future Work

The three proposed models doesn't work as well as we have expected. Although their evaluation metrics have surpassed the baseline model, the improvement is not large. In additional to the points we already mentioned in the discussion section, we can also take following steps to continue working on it to obtain potentially better performance.

c Segmentation and Edge Prediction of Model 2



d Segmentation and Edge Prediction of Model 3

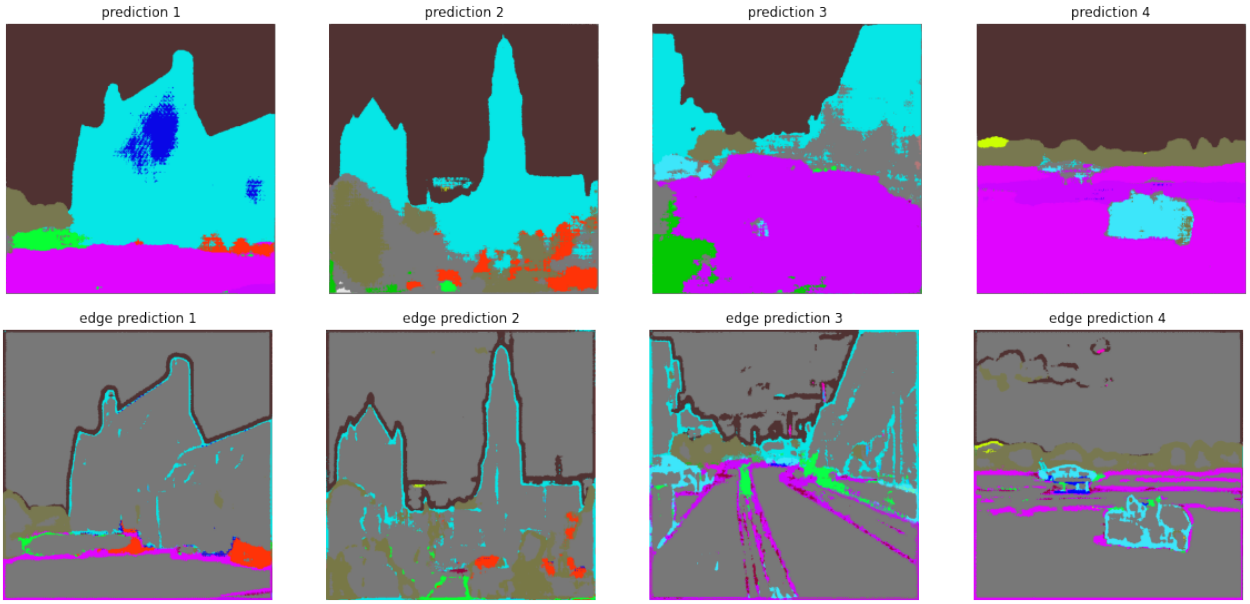


Figure 8. Segmentation Prediction Visualization

6.1. Fine-tuning

Given the strict time and computation limit, we don't have the opportunity to fine-tune critical model hyper-parameters such as learning rate and loss weights. For other components such as the optimizer and loss function, there is also room for further fine-tuning. Training a model requires more than 10 hours in Google Colab, therefore we have randomly chosen those parameters. The decoupled SegNet method [7] proposed by Li et al. has also provided guidance for our parameter choice.

6.2. Larger Dataset

Currently our model is trained for 25 epochs with $1/2$ of the whole ADE20K dataset. Our experiment with U-Net model has shown that larger dataset could help with model accuracy and reducing overfitting. Larger models relies

on larger dataset to get better performance as well. The comparison between models could also be different given different size of the training dataset. In the future, we would like to utilize more computation resources to further compare the U-Net baseline model and our proposed edge supervised model on the whole ADE20K dataset.

6.3. Longer Training and Distributed Training

Due to the resource limitation, we have trained those models for 30 epochs. At the end of training, some of the models still have a decreasing trend for loss. The SegNet paper [7] indicates that without enough training time (around one week) the decoupled model may not have good performance. Since the dataset is large, we may also consider using TPU for distributed training to reduce training time in the future.

6.4. Better Edge Supervision Architecture

In this project, we have proposed three architectures to combine edge supervision with the segmentation model, but there are still many possible ways to improve the quality of edge information and the concatenation of edge features and image features. In the future, we would like to explore the influence of different encoders/decoders, filter structures, and other parts of the model architecture.

References

[1] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, and Senior Member. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016. 1, 2

[2] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4545–4554, 2016. 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 2

[4] Alexey Dosovitskiy, Lucas Beyer, and Alexander Kolesnikov. An image is worth 16x16 words: Transformer for image recognition at scale. 2021. The paper is currently under peer review. 2

[5] Ali Hatamizadeh, Yucheng Tang, and Vishwesh Nath. Unetr: Transformers for 3d medical image segmentation. 2021. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2015. 2

[7] Xiangtai Li, Xia Li, Li Zhang, Guangliang Cheng, Jianping Shi, Zhouchen Lin, Shaohua Tan, and Yunhai Tong. Improving semantic segmentation via decoupled body and edge supervision. In *European Conference on Computer Vision*, pages 435–452. Springer, 2020. 1, 3, 11, 12

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015. 1, 2

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, 9351, 2015. 1, 2, 3

[10] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. 2008. 1, 2

[11] Paul Sturgess, Karteek Alahari, and L'ubor Ladický. Combining appearance and structure from motion features for road scene understanding. 2009. 1, 2

[12] Ashish Vaswani, Noam Shazeer, and Niki Parmar. Attention is all you need. 2017. 2

[13] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 1